SingleLayerPerceptron

March 5, 2021

0.1 CIS 666: Assignemnt 4 - Single Layer Perceptron

Bradley Dowling

CSUID: 2657649

Spring 2021

[2]:
```
import slp
```

0.2 Overview

The fully connected single layer perceptron is implemented as a class in slp.py. Running slp.py as a script will run all four tasks. Each task will block until the matplotlib displays are closed.

0.2.1 Characteristics of SLP

- The single layer perceptron can be initialized with no arguments with default settings:
  - 500 random training images
  - 100 random testing images
  - 20 iterations
  - learning rate 0.01
  - randomized bias (-1, 1)
  - epsilon 1.0e-7
  - Sigmoid activation function
  - MNIST digits dataset
- The single layer perceptron class accepts several parameters:
  - Training sets of images and labels
  - Testing sets of images and labels
  - A number of random samples from mnist to take for training and/or testing.
  - Dimensions of the input data
  - Learning rate
  - Bias
  - Binary image threshold
  - epsilon
  - activation function (pass a function or lambda)
  - which EMNIST dataset to use
  - whether to use the complete dataset instead of a subset

- The Single Layer Perceptron has been extended to support every EMNIST dataset: digits, letters, byclass, bymerge, balanced

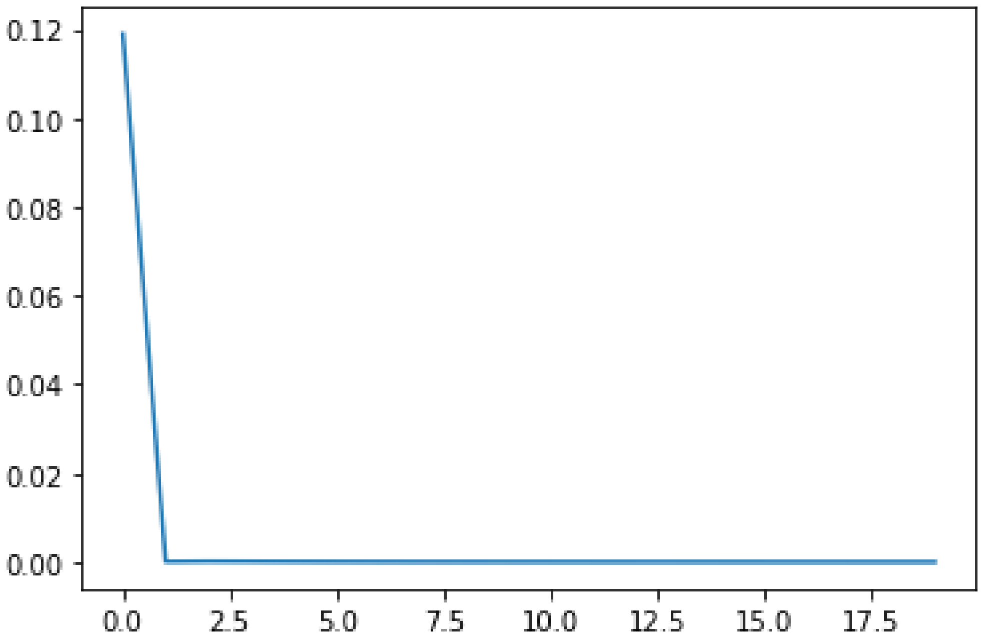
0.3 Task 0:

Design a fully connected Single Layer Perceptron with 500 training images and 100 testing images. Plot MSE vs. Iterations and Percentage error. Using binary image threshold of 0.5 (applied after normalization, so approximately 127)

[6]:

```
SLP = slp.SingleLayerPerceptron(threshold=0.5)
SLP.train()
SLP.plot_mse()
```

```
MSE iteration 0: 0.11900618821609539
MSE iteration 1: 8.826823863147954e-05
MSE iteration 2: 0.00014121513060101547
MSE iteration 3: 0.00011580188069028459
MSE iteration 4: 8.691019274881714e-05
MSE iteration 5: 6.554923690889878e-05
MSE iteration 6: 5.088646079172863e-05
MSE iteration 7: 4.140484192294091e-05
MSE iteration 8: 3.489564634916909e-05
MSE iteration 9: 2.9894757648676245e-05
MSE iteration 10: 2.5861826223961032e-05
MSE iteration 11: 2.2589965506506268e-05
MSE iteration 12: 1.991579126973292e-05
MSE iteration 13: 1.7695153200348145e-05
MSE iteration 14: 1.5808144698702637e-05
MSE iteration 15: 1.416949672537455e-05
MSE iteration 16: 1.2729691872718262e-05
MSE iteration 17: 1.1462660103717953e-05
MSE iteration 18: 1.0351747727865342e-05
MSE iteration 19: 9.38142348187047e-06
```



[7]:

```
SLP.test()
```

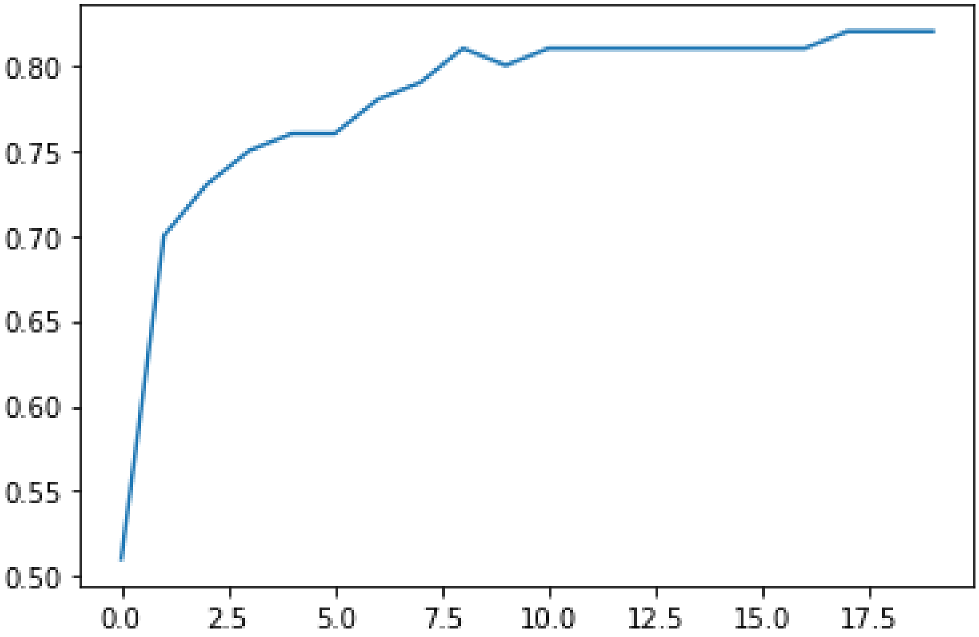
[7]:

```
0.81
```

[8]:

```
SLP.reset()
SLP.plot_accuracy()
```

```
Accuracy at iteration 0: 0.51
Accuracy at iteration 1: 0.7
Accuracy at iteration 2: 0.73
Accuracy at iteration 3: 0.75
Accuracy at iteration 4: 0.76
Accuracy at iteration 5: 0.76
Accuracy at iteration 6: 0.78
Accuracy at iteration 7: 0.79
Accuracy at iteration 8: 0.81
Accuracy at iteration 9: 0.8
Accuracy at iteration 10: 0.81
Accuracy at iteration 11: 0.81
Accuracy at iteration 12: 0.81
Accuracy at iteration 13: 0.81
Accuracy at iteration 14: 0.81
Accuracy at iteration 15: 0.81
Accuracy at iteration 16: 0.81
Accuracy at iteration 17: 0.82
Accuracy at iteration 18: 0.82
Accuracy at iteration 19: 0.82
```



With these default settings, the MSE drops and accuracy rises with each application of the entire training set.

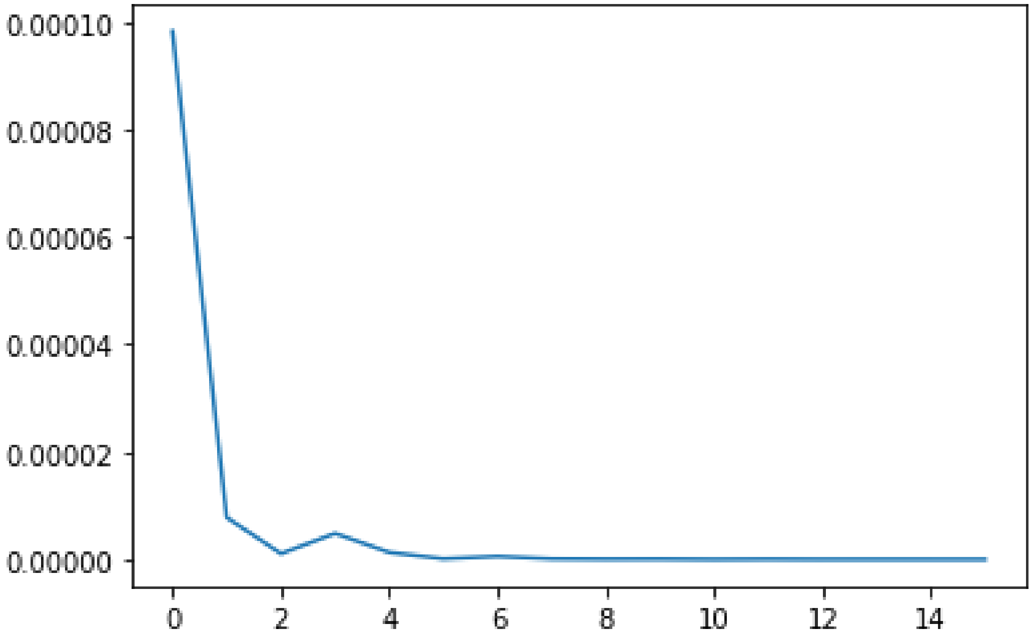
0.4 Task 1:

Repeat the first experiment with 3 learning rates. Using binary image threshold 100

[10]:

```
print("Learning rate == 1")
SLP.learning_rate = 1.0
SLP.reset()
SLP.plot_mse()
```

```
Learning rate == 1
MSE iteration 0: 9.831204383251909e-05
MSE iteration 1: 7.820061325840703e-06
MSE iteration 2: 1.0687716129174065e-06
MSE iteration 3: 4.842490426163577e-06
MSE iteration 4: 1.3014018968353643e-06
MSE iteration 5: 1.1449905395626226e-07
MSE iteration 6: 5.5694259684834e-07
MSE iteration 7: 1.0097194136115652e-07
MSE iteration 8: 4.877579081316188e-08
MSE iteration 9: 5.9386560976087026e-08
MSE iteration 10: 1.8671685246986748e-10
MSE iteration 11: 3.4136914674650336e-08
MSE iteration 12: 1.949748910811996e-09
MSE iteration 13: 5.5760860404519194e-09
MSE iteration 14: 3.422729042084748e-10
MSE iteration 15: 1.4120880558054999e-12
Reached epsilon 1e-10 at 15 iterations
```

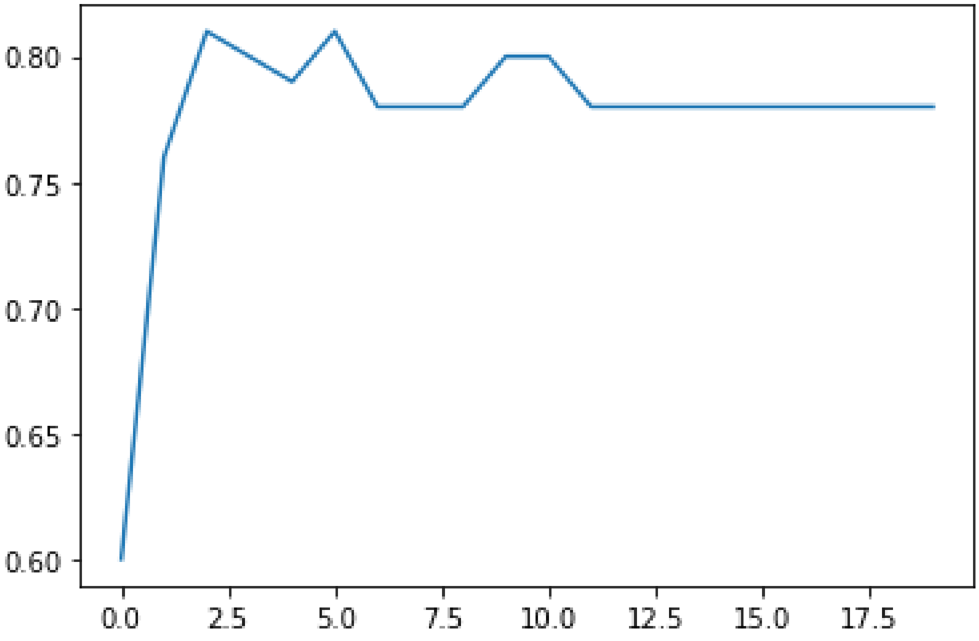


[11]:
```
SLP.test()
```

[11]: 0.77

[12]:
```
SLP.reset()
SLP.plot_accuracy()
```

```
Accuracy at iteration 0: 0.6
Accuracy at iteration 1: 0.76
Accuracy at iteration 2: 0.81
Accuracy at iteration 3: 0.8
Accuracy at iteration 4: 0.79
Accuracy at iteration 5: 0.81
Accuracy at iteration 6: 0.78
Accuracy at iteration 7: 0.78
Accuracy at iteration 8: 0.78
Accuracy at iteration 9: 0.8
Accuracy at iteration 10: 0.8
Accuracy at iteration 11: 0.78
Accuracy at iteration 12: 0.78
Accuracy at iteration 13: 0.78
Accuracy at iteration 14: 0.78
Accuracy at iteration 15: 0.78
Accuracy at iteration 16: 0.78
Accuracy at iteration 17: 0.78
Accuracy at iteration 18: 0.78
Accuracy at iteration 19: 0.78
```

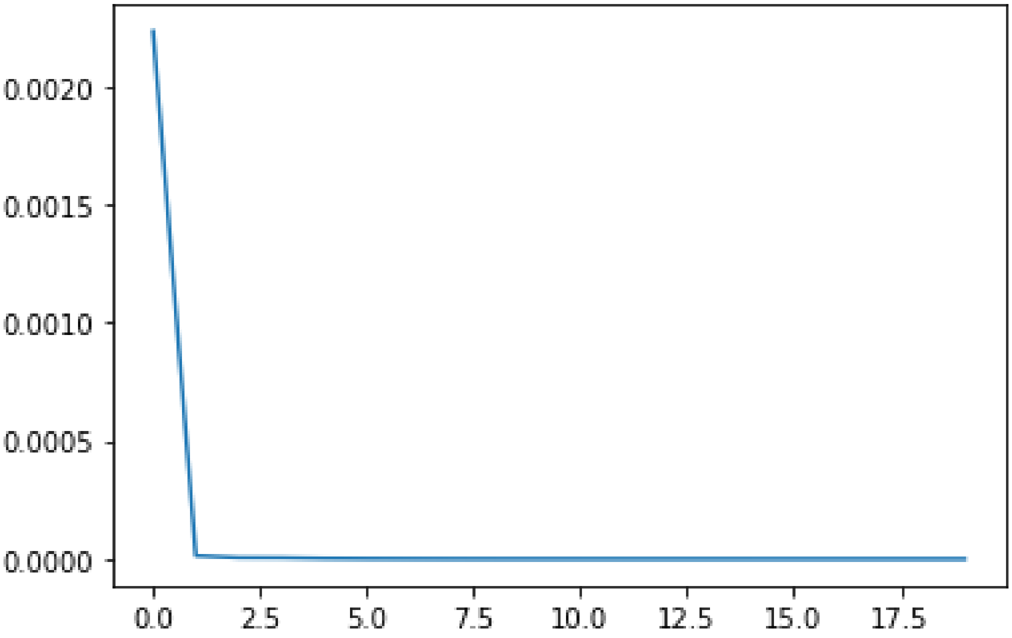


Performance is more erratic and less accurate with a large learning rate.

[13]:

```
print("Learning rate 0.1")
SLP.learning_rate = 0.1
SLP.reset()
SLP.plot_mse()
```

```
Learning rate 0.1
MSE iteration 0: 0.00223093198392172
MSE iteration 1: 1.3755340953855757e-05
MSE iteration 2: 5.715690052489872e-06
MSE iteration 3: 5.096811793907744e-06
MSE iteration 4: 2.40430459730923e-06
MSE iteration 5: 1.2854971862477071e-06
MSE iteration 6: 7.528902853991178e-07
MSE iteration 7: 3.682750712872206e-07
MSE iteration 8: 2.507781181948531e-07
MSE iteration 9: 1.9338676502584478e-07
MSE iteration 10: 1.5790116029424572e-07
MSE iteration 11: 1.329364573297691e-07
MSE iteration 12: 1.1396405455190443e-07
MSE iteration 13: 9.89830382017881e-08
MSE iteration 14: 8.687977190107952e-08
MSE iteration 15: 7.693910953843669e-08
MSE iteration 16: 6.866284916003166e-08
MSE iteration 17: 6.16897110205673e-08
MSE iteration 18: 5.575199662268593e-08
MSE iteration 19: 5.064831455851943e-08
```

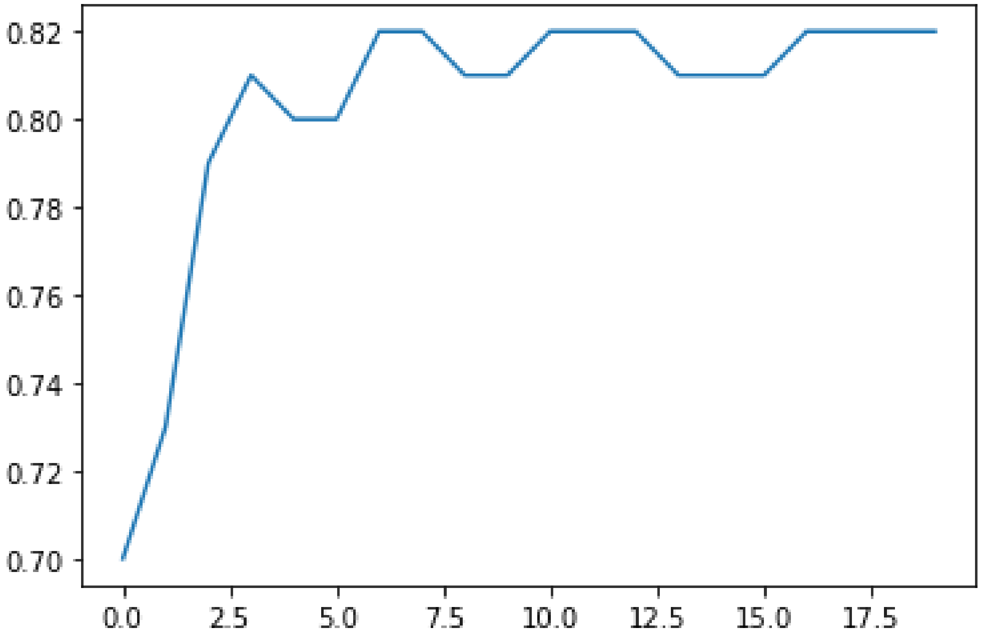


[14]:
```
SLP.test()
```

[14]:
```
0.83
```

[15]:
```
SLP.reset()
SLP.plot_accuracy()
```

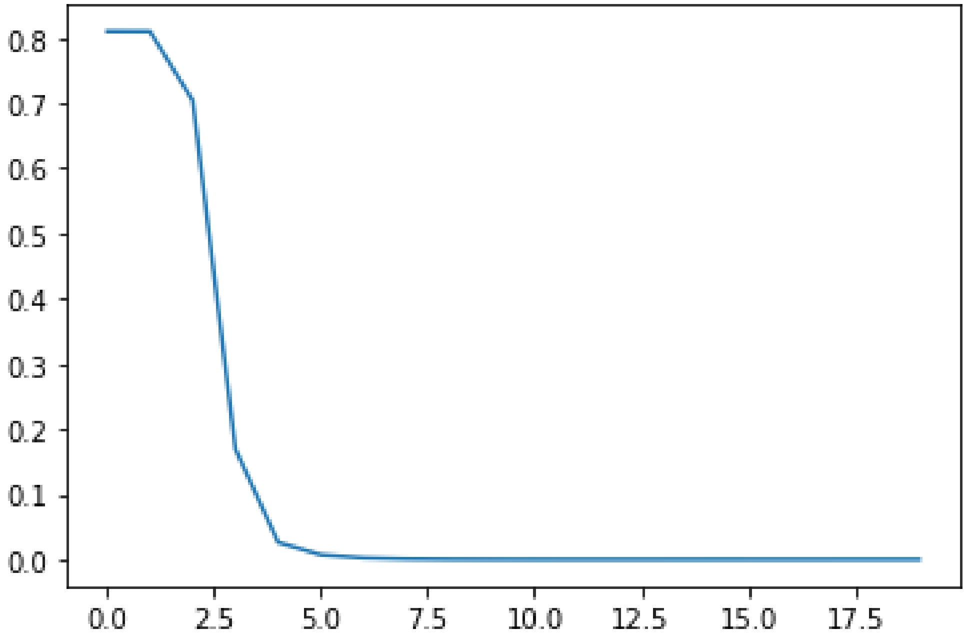
```
Accuracy at iteration 0: 0.7
Accuracy at iteration 1: 0.73
Accuracy at iteration 2: 0.79
Accuracy at iteration 3: 0.81
Accuracy at iteration 4: 0.8
Accuracy at iteration 5: 0.8
Accuracy at iteration 6: 0.82
Accuracy at iteration 7: 0.82
Accuracy at iteration 8: 0.81
Accuracy at iteration 9: 0.81
Accuracy at iteration 10: 0.82
Accuracy at iteration 11: 0.82
Accuracy at iteration 12: 0.82
Accuracy at iteration 13: 0.81
Accuracy at iteration 14: 0.81
Accuracy at iteration 15: 0.81
Accuracy at iteration 16: 0.82
Accuracy at iteration 17: 0.82
Accuracy at iteration 18: 0.82
Accuracy at iteration 19: 0.82
```



Performance is more stable, and does not end lower than it's highest point as in the first example.

[18]:
```
print("Learning rate == 0.001")
SLP.learning_rate = 0.001
SLP.reset()
SLP.plot_mse()
```

```
Learning rate == 0.001
MSE iteration 0: 0.8099999542766764
MSE iteration 1: 0.8098008435978724
MSE iteration 2: 0.7040751664294627
MSE iteration 3: 0.16951182845006907
MSE iteration 4: 0.026255891601836365
MSE iteration 5: 0.007179673977473417
MSE iteration 6: 0.002419843640658535
MSE iteration 7: 0.0009020014782858893
MSE iteration 8: 0.00038460616805876174
MSE iteration 9: 0.00019910755107650703
MSE iteration 10: 0.00013293479483295999
MSE iteration 11: 0.00011429050508374558
MSE iteration 12: 0.00011683157534953106
MSE iteration 13: 0.00012942897346491004
MSE iteration 14: 0.00014561322764401548
MSE iteration 15: 0.00016046380783387367
MSE iteration 16: 0.00017136837582495756
MSE iteration 17: 0.00017779152894768728
MSE iteration 18: 0.00018052370514341694
MSE iteration 19: 0.00018075801654392773
```

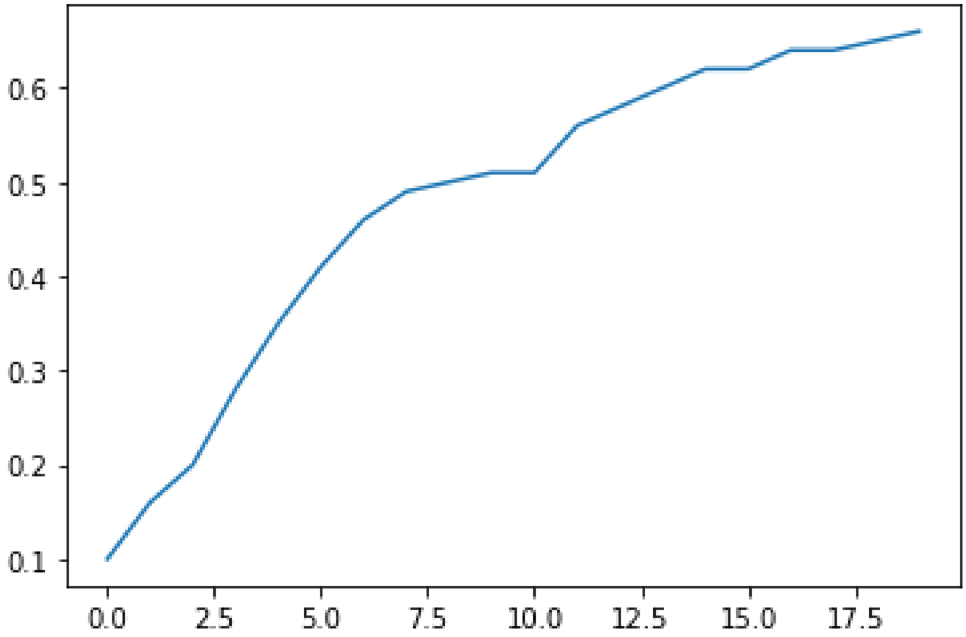


[19]:
```
SLP.test()
```

[19]:
```
0.67
```

[21]:
```
SLP.reset()
SLP.plot_accuracy()
```

```
Accuracy at iteration 0: 0.1
Accuracy at iteration 1: 0.16
Accuracy at iteration 2: 0.2
Accuracy at iteration 3: 0.28
Accuracy at iteration 4: 0.35
Accuracy at iteration 5: 0.41
Accuracy at iteration 6: 0.46
Accuracy at iteration 7: 0.49
Accuracy at iteration 8: 0.5
Accuracy at iteration 9: 0.51
Accuracy at iteration 10: 0.51
Accuracy at iteration 11: 0.56
Accuracy at iteration 12: 0.58
Accuracy at iteration 13: 0.6
Accuracy at iteration 14: 0.62
Accuracy at iteration 15: 0.62
Accuracy at iteration 16: 0.64
Accuracy at iteration 17: 0.64
Accuracy at iteration 18: 0.65
Accuracy at iteration 19: 0.66
```



With a very, very small learning rate, the model takes too long to converge and does not perform as well. The default learning rate of 0.01 performed the best out of these tests.

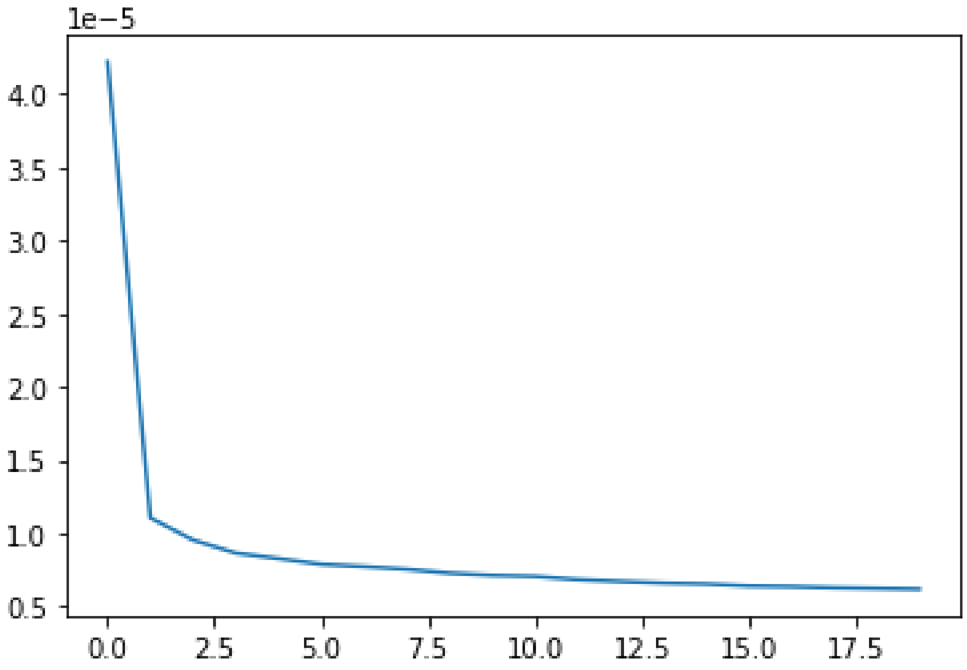
0.5 Task 2:

Repeat task 1 with 10000 training images and 1000 testing images.

```
[23]: SLP2 = slp.SingleLayerPerceptron(ntrain = 10000, ntest = 1000, threshold = 0.5)
```

[26]:
```
print("Learning rate == 0.1")
SLP2.learning_rate = 0.1
SLP2.reset()
SLP2.plot_mse()
```

```
Learning rate == 0.1
MSE iteration 0: 4.2226931096648863e-05
MSE iteration 1: 1.1058072012776823e-05
MSE iteration 2: 9.537417314542023e-06
MSE iteration 3: 8.6446727444273e-06
MSE iteration 4: 8.268885493086502e-06
MSE iteration 5: 7.878365671441897e-06
MSE iteration 6: 7.721489935359038e-06
MSE iteration 7: 7.537606293856917e-06
MSE iteration 8: 7.273056580551813e-06
MSE iteration 9: 7.121673014881158e-06
MSE iteration 10: 7.050377462475995e-06
MSE iteration 11: 6.833438559741888e-06
MSE iteration 12: 6.705507011831964e-06
MSE iteration 13: 6.595905122341934e-06
MSE iteration 14: 6.533594441821425e-06
MSE iteration 15: 6.3970930870640816e-06
MSE iteration 16: 6.355860582357641e-06
MSE iteration 17: 6.275170111455369e-06
MSE iteration 18: 6.230790375716839e-06
MSE iteration 19: 6.18035188039528e-06
```



[27]:

```
SLP2.test()
```

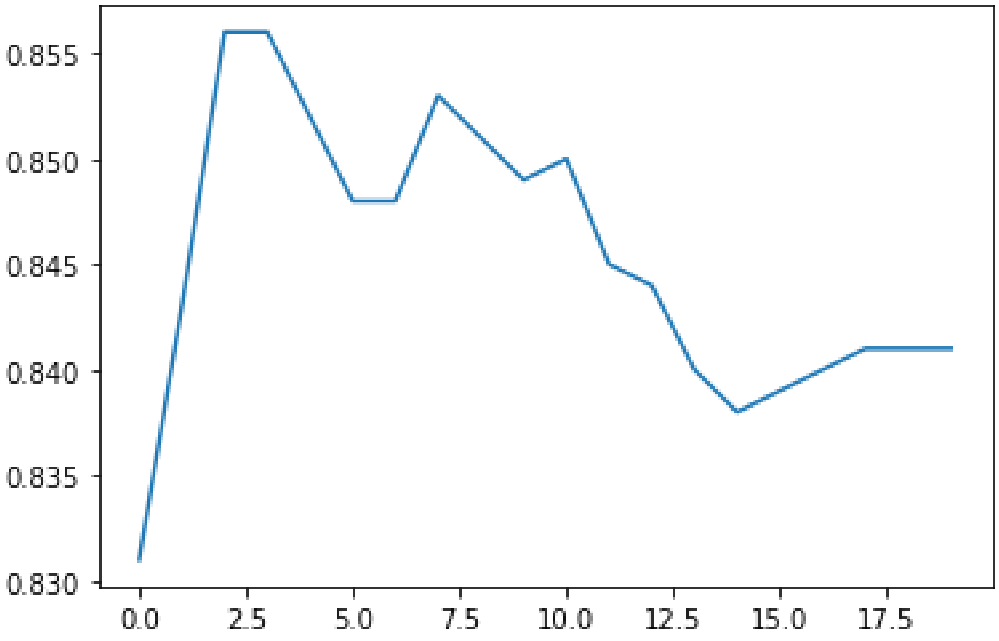
[27]:

```
0.84
```

[28]:

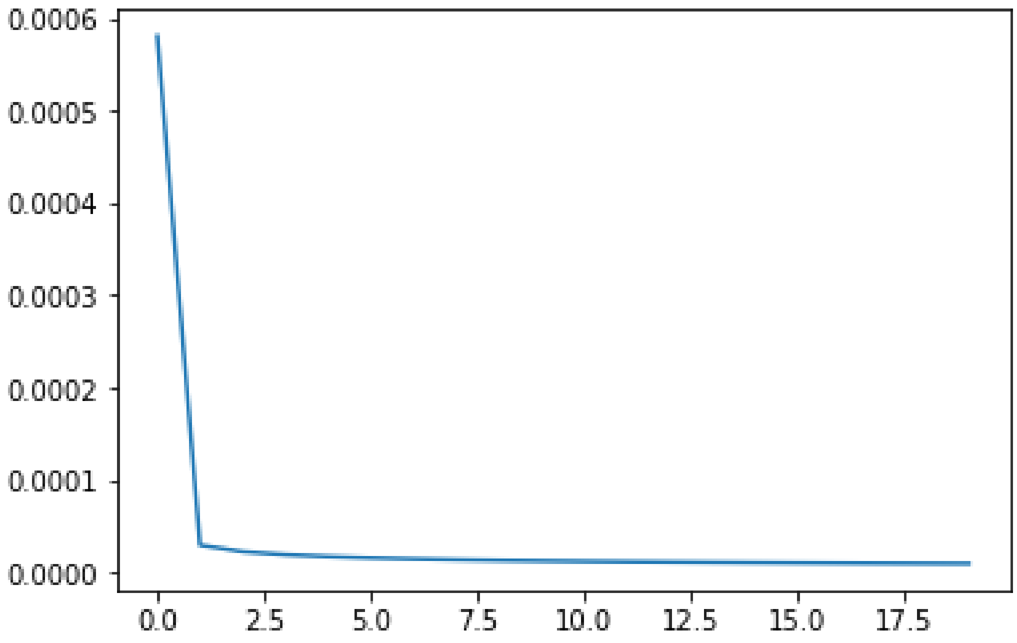
```
SLP2.reset()
SLP2.plot_accuracy()
```

```
Accuracy at iteration 0: 0.831
Accuracy at iteration 1: 0.843
Accuracy at iteration 2: 0.856
Accuracy at iteration 3: 0.856
Accuracy at iteration 4: 0.852
Accuracy at iteration 5: 0.848
Accuracy at iteration 6: 0.848
Accuracy at iteration 7: 0.853
Accuracy at iteration 8: 0.851
Accuracy at iteration 9: 0.849
Accuracy at iteration 10: 0.85
Accuracy at iteration 11: 0.845
Accuracy at iteration 12: 0.844
Accuracy at iteration 13: 0.84
Accuracy at iteration 14: 0.838
Accuracy at iteration 15: 0.839
Accuracy at iteration 16: 0.84
Accuracy at iteration 17: 0.841
Accuracy at iteration 18: 0.841
Accuracy at iteration 19: 0.841
```



[32]:
```
print("Learning rate == 0.01")
SLP2.learning_rate = 0.01
SLP2.reset()
SLP2.plot_mse()
```

```
Learning rate == 0.01
MSE iteration 0: 0.0005811355413191082
MSE iteration 1: 2.9571476670276213e-05
MSE iteration 2: 2.2503780771739445e-05
MSE iteration 3: 1.919027940939956e-05
MSE iteration 4: 1.715382924087223e-05
MSE iteration 5: 1.5725247885447967e-05
MSE iteration 6: 1.4647467966497493e-05
MSE iteration 7: 1.3797370804958482e-05
MSE iteration 8: 1.310560325339195e-05
MSE iteration 9: 1.2529001194839134e-05
MSE iteration 10: 1.203908088444156e-05
MSE iteration 11: 1.1616283550162905e-05
MSE iteration 12: 1.1246728039855433e-05
MSE iteration 13: 1.0920257320032165e-05
MSE iteration 14: 1.0629234308045472e-05
MSE iteration 15: 1.0367786835837614e-05
MSE iteration 16: 1.0131317839662436e-05
MSE iteration 17: 9.916173814435753e-06
MSE iteration 18: 9.71941188221269e-06
MSE iteration 19: 9.53863174036985e-06
```



[33]:

```
SLP2.test()
```

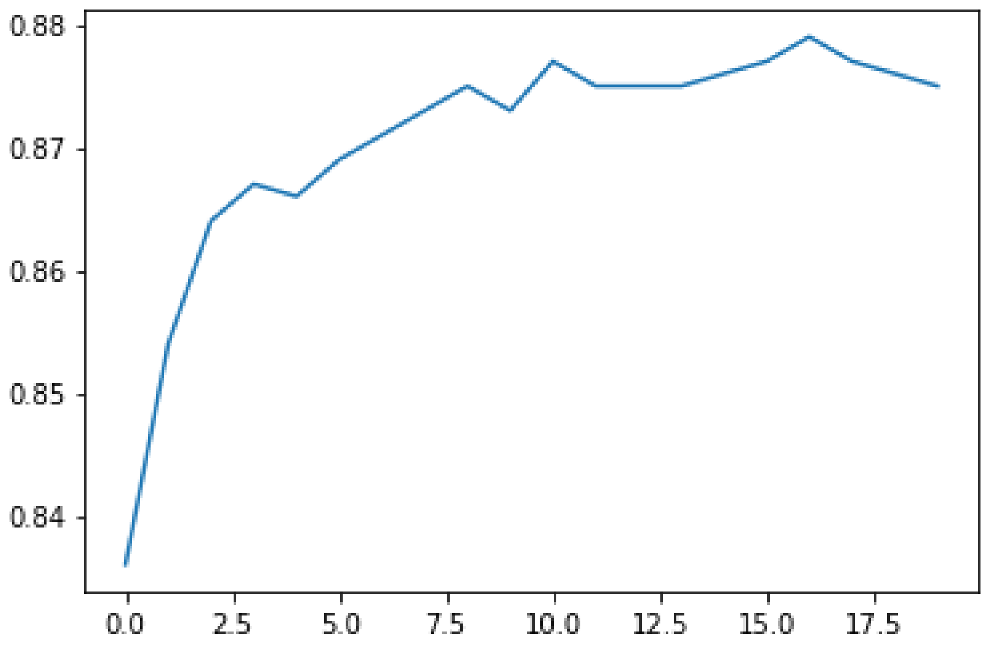
[33]:

```
0.877
```

[34]:

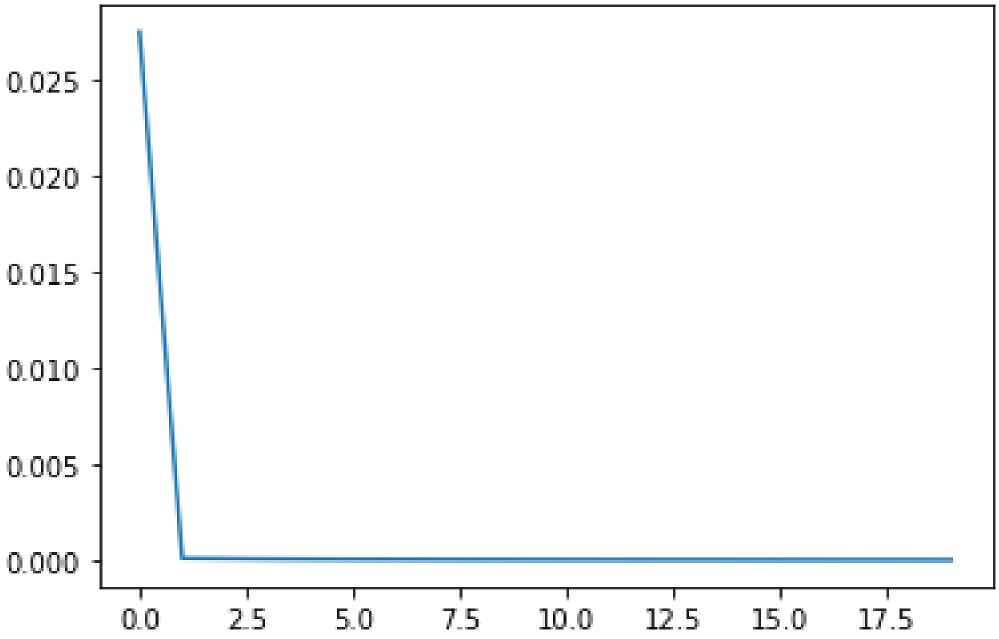
```
SLP2.reset()
SLP2.plot_accuracy()
```

```
Accuracy at iteration 0: 0.836
Accuracy at iteration 1: 0.854
Accuracy at iteration 2: 0.864
Accuracy at iteration 3: 0.867
Accuracy at iteration 4: 0.866
Accuracy at iteration 5: 0.869
Accuracy at iteration 6: 0.871
Accuracy at iteration 7: 0.873
Accuracy at iteration 8: 0.875
Accuracy at iteration 9: 0.873
Accuracy at iteration 10: 0.877
Accuracy at iteration 11: 0.875
Accuracy at iteration 12: 0.875
Accuracy at iteration 13: 0.875
Accuracy at iteration 14: 0.876
Accuracy at iteration 15: 0.877
Accuracy at iteration 16: 0.879
Accuracy at iteration 17: 0.877
Accuracy at iteration 18: 0.876
Accuracy at iteration 19: 0.875
```



[35]:
```
print("Learning rate == 0.001")
SLP2.learning_rate = 0.001
SLP2.reset()
SLP2.plot_mse()
```

```
Learning rate == 0.001
MSE iteration 0: 0.027522258732161436
MSE iteration 1: 0.00010966391239601831
MSE iteration 2: 9.041419091126196e-05
MSE iteration 3: 7.25565782842762e-05
MSE iteration 4: 6.121083960172636e-05
MSE iteration 5: 5.3605838074692726e-05
MSE iteration 6: 4.815401541520709e-05
MSE iteration 7: 4.4026623704442336e-05
MSE iteration 8: 4.076738556426323e-05
MSE iteration 9: 3.811166494931181e-05
MSE iteration 10: 3.5897121523809027e-05
MSE iteration 11: 3.401769573306736e-05
MSE iteration 12: 3.2399928139203086e-05
MSE iteration 13: 3.09907734507189e-05
MSE iteration 14: 2.975083263572264e-05
MSE iteration 15: 2.8650207110905183e-05
MSE iteration 16: 2.76657887879412e-05
MSE iteration 17: 2.6779424183536473e-05
MSE iteration 18: 2.597663949177871e-05
MSE iteration 19: 2.524573521001014e-05
```

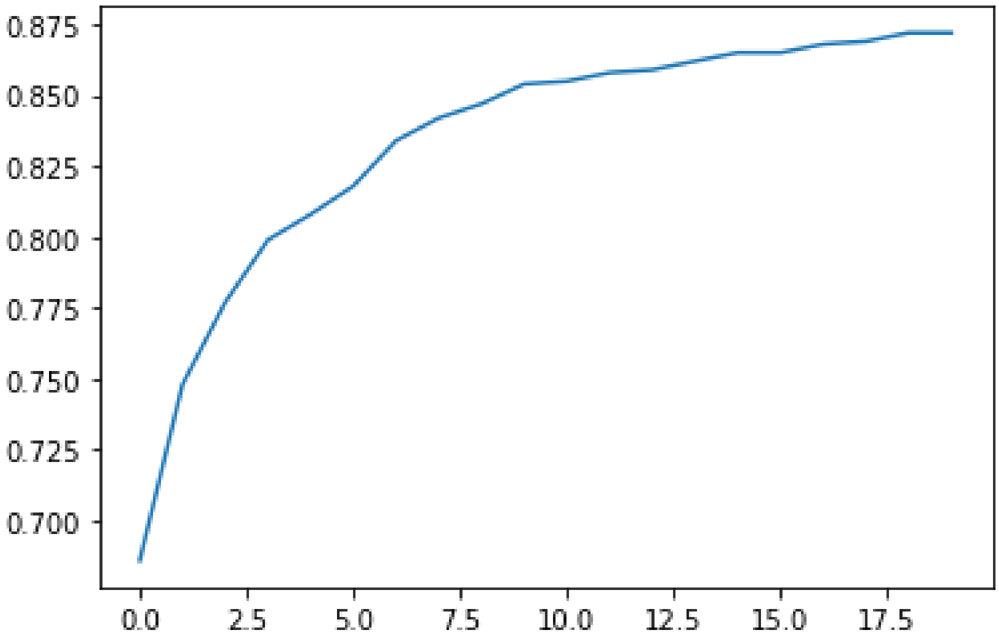


```
[36]: SLP2.test()
```

```
[36]: 0.863
```

```
[37]: SLP2.reset()
      SLP2.plot_accuracy()
```

```
Accuracy at iteration 0: 0.686
Accuracy at iteration 1: 0.748
Accuracy at iteration 2: 0.777
Accuracy at iteration 3: 0.799
Accuracy at iteration 4: 0.808
Accuracy at iteration 5: 0.818
Accuracy at iteration 6: 0.834
Accuracy at iteration 7: 0.842
Accuracy at iteration 8: 0.847
Accuracy at iteration 9: 0.854
Accuracy at iteration 10: 0.855
Accuracy at iteration 11: 0.858
Accuracy at iteration 12: 0.859
Accuracy at iteration 13: 0.862
Accuracy at iteration 14: 0.865
Accuracy at iteration 15: 0.865
Accuracy at iteration 16: 0.868
Accuracy at iteration 17: 0.869
Accuracy at iteration 18: 0.872
Accuracy at iteration 19: 0.872
```



Increasing the number of testing images improved performance, and 0.01 learning rate still performed the best with 20 iterations.

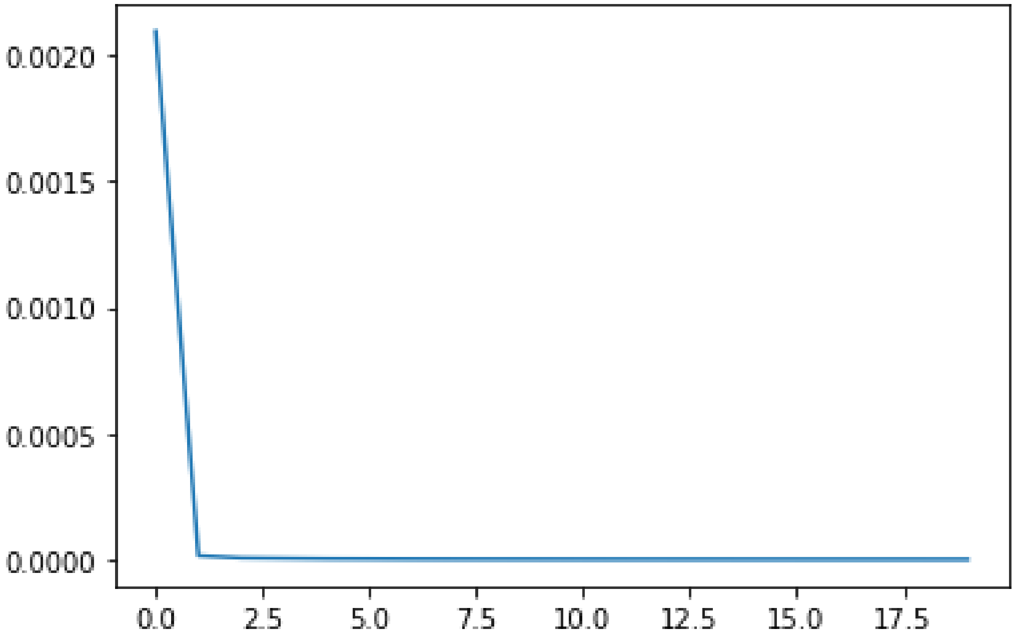
0.6 Task 3:

Repeat task 1 without using binary image thresholding.

[39]:
```
SLP3 = slp.SingleLayerPerceptron()
```

[40]:
```
print("Learning rate == 0.001")
SLP3.learning_rate = 0.1
SLP3.reset()
SLP3.plot_mse()
```

```
Learning rate == 0.001
MSE iteration 0: 0.002093801008132573
MSE iteration 1: 1.5174984551692087e-05
MSE iteration 2: 5.987495630948375e-06
MSE iteration 3: 4.372596975638773e-06
MSE iteration 4: 2.792335352596284e-06
MSE iteration 5: 2.180744704095797e-06
MSE iteration 6: 1.503886923845236e-06
MSE iteration 7: 1.1521224516929976e-06
MSE iteration 8: 9.486939106061179e-07
MSE iteration 9: 7.740337149481379e-07
MSE iteration 10: 6.418515986487952e-07
MSE iteration 11: 5.508043714629911e-07
MSE iteration 12: 4.715338498590076e-07
MSE iteration 13: 4.009528518383896e-07
MSE iteration 14: 3.4285452769946746e-07
MSE iteration 15: 2.9902915738369964e-07
MSE iteration 16: 2.65013825130067e-07
MSE iteration 17: 2.3601264539390218e-07
MSE iteration 18: 2.1010389480429082e-07
MSE iteration 19: 1.8724006313992001e-07
```

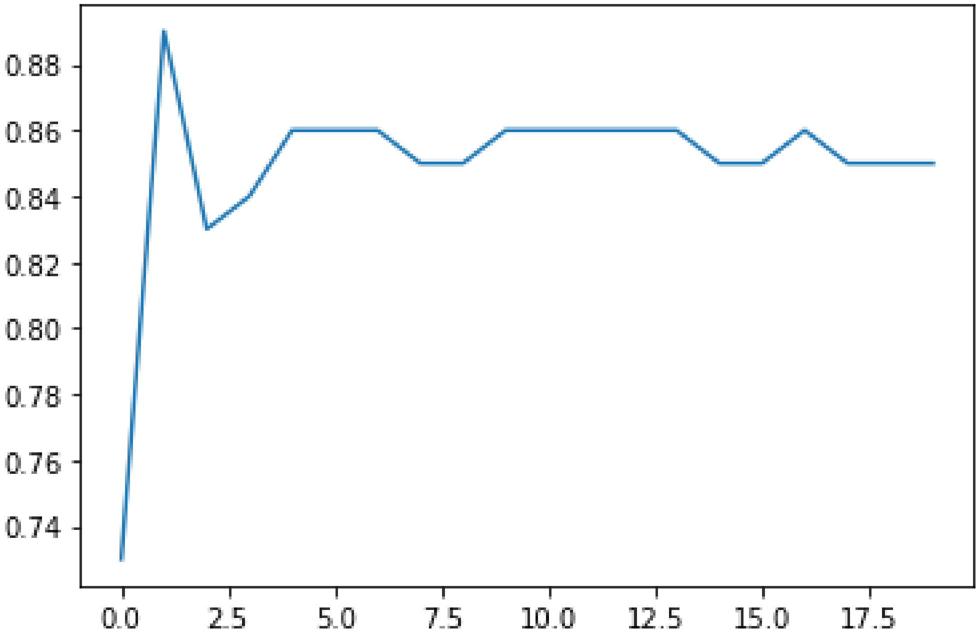


[41]:
```
SLP3.test()
```

[41]: 0.84

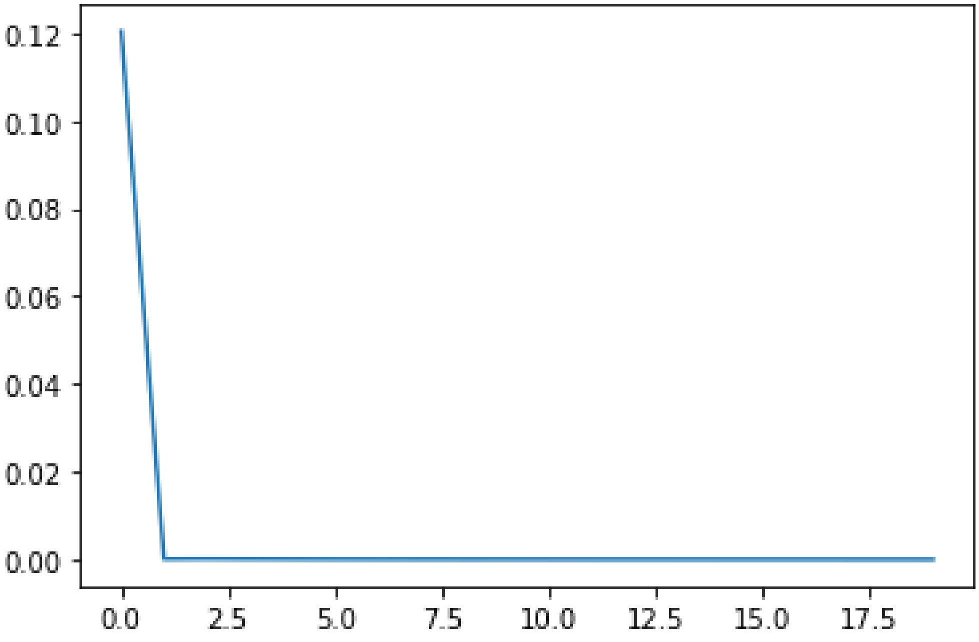
[43]:
```
SLP3.reset()
SLP3.plot_accuracy()
```

```
Accuracy at iteration 0: 0.73
Accuracy at iteration 1: 0.89
Accuracy at iteration 2: 0.83
Accuracy at iteration 3: 0.84
Accuracy at iteration 4: 0.86
Accuracy at iteration 5: 0.86
Accuracy at iteration 6: 0.86
Accuracy at iteration 7: 0.85
Accuracy at iteration 8: 0.85
Accuracy at iteration 9: 0.86
Accuracy at iteration 10: 0.86
Accuracy at iteration 11: 0.86
Accuracy at iteration 12: 0.86
Accuracy at iteration 13: 0.86
Accuracy at iteration 14: 0.85
Accuracy at iteration 15: 0.85
Accuracy at iteration 16: 0.86
Accuracy at iteration 17: 0.85
Accuracy at iteration 18: 0.85
Accuracy at iteration 19: 0.85
```



[44]:
```
print("Learning rate == 0.01")
SLP3.learning_rate = 0.01
SLP3.reset()
SLP3.plot_mse()
```

```
Learning rate == 0.001
MSE iteration 0: 0.12036858070344392
MSE iteration 1: 0.0001077902598293652
MSE iteration 2: 0.0001470217914924507
MSE iteration 3: 0.00011051566708550121
MSE iteration 4: 8.579572625920981e-05
MSE iteration 5: 6.709840443325815e-05
MSE iteration 6: 5.385276954358709e-05
MSE iteration 7: 4.405807645377951e-05
MSE iteration 8: 3.663362234407749e-05
MSE iteration 9: 3.111912022131567e-05
MSE iteration 10: 2.699149724088988e-05
MSE iteration 11: 2.3797469714571284e-05
MSE iteration 12: 2.1232745185447927e-05
MSE iteration 13: 1.9111125552669e-05
MSE iteration 14: 1.7320396343051017e-05
MSE iteration 15: 1.5788701594346498e-05
MSE iteration 16: 1.4465719713476907e-05
MSE iteration 17: 1.3314248718305503e-05
MSE iteration 18: 1.2305729494149071e-05
MSE iteration 19: 1.1417395856118204e-05
```



[45]:
```
SLP3.test()
```

[45]:
```
0.88
```

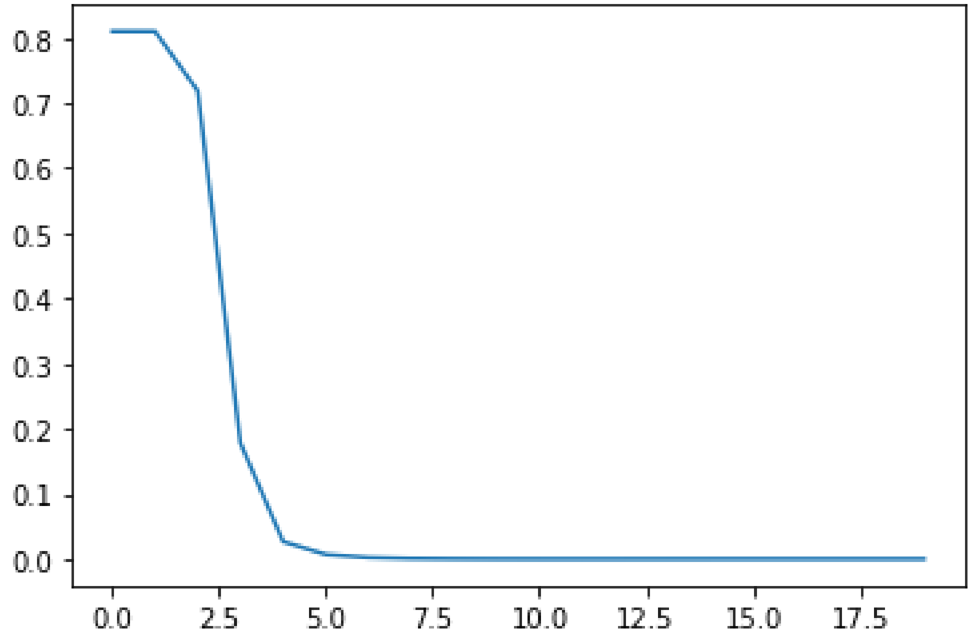
[46]:
```
SLP3.reset()
SLP3.plot_accuracy
```

[46]:
```
<bound method SingleLayerPerceptron.plot_accuracy of <slp.SingleLayerPerceptron
object at 0x7f3dd2c01430>>
```

[47]:
```
print("Learning rate == 0.001")
SLP3.learning_rate = 0.001
```

```
SLP3.reset()
SLP3.plot_mse()
```

```
Learning rate == 0.001
MSE iteration 0: 0.8099999417298823
MSE iteration 1: 0.8098904390345136
MSE iteration 2: 0.7189531031522729
MSE iteration 3: 0.1781975172198422
MSE iteration 4: 0.02671266502839634
MSE iteration 5: 0.006954941716668496
MSE iteration 6: 0.0023129410369105334
MSE iteration 7: 0.0008647411151733232
MSE iteration 8: 0.00037300486521588166
MSE iteration 9: 0.00019500011683178976
MSE iteration 10: 0.00013023067800448814
MSE iteration 11: 0.00011419993199758301
MSE iteration 12: 0.00012056583648949373
MSE iteration 13: 0.00013536557356152068
MSE iteration 14: 0.00015089348868081444
MSE iteration 15: 0.00016367872251370663
MSE iteration 16: 0.00017290632087886494
MSE iteration 17: 0.00017876855023878554
MSE iteration 18: 0.00018171265650320073
MSE iteration 19: 0.00018229150141575766
```



[48]:
```
SLP3.test()
```

```
[48]: 0.73
```

```
[ ]: SLP3.reset()
     SLP3.plot_accuracy()
```

Results using binary images are broadly comparable to using the multilevel data, and results are similar across learning rates.

0.7 Task 4:

Compare accuracy to that of the SVM from Assignment 3.

The best accuracy in these trials was 87.7%. In other tests, the best result I was able to attain was 93.5%. The variation in results could be attributed to different randomized weight starting points, or suboptimal choices in hyperparameter values.

The SVM performance was better on average, with a maximum performance of 95%, and average performance of about 93%.

0.8 Conclusions

Many factors can influence the performance of this simple neural network, and learning rate is one of the most impactful. Testing with other activation functions yielded very different results as well. In general, sigmoid performed the best, followed by ReLU, and linear, and heaviside perfomed fairly well. The Hypertangent activation function does not tend to perform well with this model.

Accuracy tended to be similar across the other datasets available in the MNIST dataset. By and large, a learning rate in the 0.001 - 0.01 tended to perform the best, where the smaller rate could be very effective given a large number of iterations.

The SingleLayerPerceptron class provides many different options, and it is interesting to run it against the other data sets in EMNIST. The largest group has 61 classes of characters, and the performance tends to be similar as with digits. There are many options available in this class, and there are function objects in the source file for sigmoid, relu, tanh, linear, and heaviside activation functions. I did not have time to create a nice user interface yet, but I'll update the code on https://github.com/hairshirt/single-layer-perceptron with an argparse tui.